JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 27 No. 7, pp. 65-73, July 2022
https://doi.org/10.9708/jksci.2022.27.07.065

Performance Comparison and Analysis of Container-based Host Operating Systems for sending and receiving High-capacity data on Server Systems

Sungho Kim*, Oeon Kwon*, Jung Han Kim*, JiHyeon Byeon*, Sang-Ho Hwang*

*Researcher, Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology, Gyeongsan, Korea
*Researcher, Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology, Gyeongsan, Korea
*Researcher, Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology, Gyeongsan, Korea
*Researcher, Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology, Gyeongsan, Korea
*Researcher, Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology, Gyeongsan, Korea

[Abstract]

Recently, as the Windows system supports the Windows subsystem for Linux (WSL), various researchers have studied to apply a docker container on various systems such as server systems, workstation system and so on. However, in various existing researchers, there is a lack of performance-related indicators to apply the system to each operating system (linux system and windows system). In this paper, we compared a performance comparison and analysis of container-based host operating systems. We configured experimental environments of operating systems for microsoft windows systems and linux systems based on a docker container support. In experimental results, the containers of linux systems reduced the average data latency of dataset 1-6 by 3.9%, 62.16%, 1552.38%, 7.27%, 60.83%, and 1567.2%, compared to the containers on microsoft windows systems.

▸ **Key words**: High-capacity data, Performance comparison, Lightweight, Docker container, Digital twin, Smart factory

• First Author: Sungho Kim, Corresponding Author: Sang-Ho Hwang
*Sungho Kim (shk@gitc.or.kr), Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology
*Oeon Kwon (oekwon@gitc.or.kr), Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology
*Jung Han Kim (jhkim@gitc.or.kr), Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology
*JiHyeon Byeon (jhbyeon@gitc.or.kr), Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology
*Sang-Ho Hwang (shhwang@gitc.or.kr), Dept. of Research Development, Gyeongbuk Institute of IT Convergence Industry Technology
• Received: 2022. 05. 31, Revised: 2022. 07. 07, Accepted: 2022. 07. 08.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

[요 약]

최근 윈도우 시스템에서는 리눅스용 윈도우즈 하위 시스템(Windows Subsystem for Linux, WSL)을 지원함에 따라 도커 컨테이너를 해당 시스템에 적용하기 위해 다양한 연구가 진행 중에 있다. 그러나 기존의 다양한 연구에서는 호스트 운영체제별로 시스템을 적용하기 위해 성능과 관련한 지표가 부족할 실정이다. 본 논문에서는 도커 컨테이너 기반의 호스트 운영체제별 성능 비교 분석을 진행하고자 한다. 성능 비교 분석을 진행하기 위해, 실험 환경에서는 동일한 스펙 환경과 데이터셋을 구성하였다. 실험 결과에서 리눅스 시스템에서의 도커 컨테이너가 윈도우즈 시스템 대비 데이터셋 1-6의 기준으로 3.9%, 62.16%, 1552.38%, 7.27%, 60.83%의 평균 지연시간 감소를 보였다.

▸ **주제어**: 대용량 데이터, 성능 분석, 경량화, 도커 컨테이너, 디지털 트윈, 스마트팩토리

I. Introduction

제조업 산업현장에서는 추락·끼임 등으로 인해 사망사고가 끊이질 않고 있다. 고용노동부 산하 안전보건공단에서는 2021년도 상반기 2만4천여 개 사업장을 불시 점검을 진행하였고, 2021.5.24. 보도 자료를 배포했다[1]. 보도 자료에 의하면 안전보건공단은 제조업 7,173개 사업장을 점검했으며, 이 중 3,937개 사업장에 대하여 8,102건의 위험요인을 지적했다. 특히 위험 요인 중 대다수는 고위험군 장비로 분류되는 컨베이어, 프레스, 분쇄기 등에서 끼임(2,942건)에 해당하는 위험 요인이 가장 많았으며, 뒤를 이어 떨어짐(1,872건), 부딪힘(1,277건), 화재·폭발(513건)과 같은 위험 요인으로 지적했다.

안전보건공단에서는 중대재해 위험요인 근절을 위해 제조업 중소사업장을 대상으로 3대 주요 안전조치(추락·끼임 위험 방지조치, 필수 안전보호구 착용 등) 준수 여부를 집중 점검하는 “패트롤 현장점검”을 지속적으로 실시하고 있다. 이러한 노력에도 불구하고 제조업 중소사업장에서는 2017년도에서 2020년도까지 추락·끼임 사망사고 비중이 50% 수준에서 줄어들지 않은 실정이다. 이는 단순히 현장점검으로 사망사고를 해결하기 어렵다는 것을 의미하는 것이다.

이러한 문제점을 해결하기 위해, 제조업 중소사업장에서는 정부주요정책인 ‘스마트제조’ 사업의 일환으로 산업현장 안정화 및 제조 생산력 개선을 진행하고 있는 실정이다[2]. 최근 들어 제조업 중소사업장에서는 스마트제조 기반 현장 환경을 가상으로 복제하는 기술인 디지털 트윈(Digital Twin) 기술을 확장하는 연구가 진행 중에 있다[3].

디지털 트윈 기술은 제조업 산업현장에서 방대한 양의 데이터를 가상화한 후 현실세계와 유사 혹은 동일하게 보유중인 기술로 불량원인 파악, 제조현장 상황 파악 등 제조 생산력 증대와 더불어 안전재해(부상, 사망사고 등)를 방지하기 위한 새로운 기술로 각광받고 있다. 특히 디지털 트윈 기술은 방대한 양의 센서, 데이터를 송수신하는 특성이 있기 때문에 서버 시스템을 구성에 따라 데이터 송수신에 대한 많은 양의 작업부하가 발생할 수 있다.

예를 들어 디지털 트윈 기반 스마트팩토리 환경에서는 개별 장비 별로 자이로/지자기/가속도, 온도/습도, 근접센서, 라이다, 카메라 이미지 등 다양한 정보를 수신처리를 수행한다. 이러한 데이터들은 바이트 단위에서 기가바이트 단위까지 다양하게 전송이 될 수 있어, 스마트팩토리 기반 디지털트윈 환경에서는 이러한 방대한 양에 데이터의 경우 서버 시스템에서 상당한 부하가 될 수 있다. 따라서 서버 시스템은 많은 양 데이터 송수신에 대한 작업부하를 줄이는 것이 시스템 전반에 성능 향상에 기여할 수 있다.

최근 서버 시스템에서는 경량화 어플리케이션을 활용하는 도커 컨테이너 기반 서비스 구성을 통해 시스템 전반의 성능을 향상하고자 노력중이 있다. 도커 컨테이너는 기존 가상화 시스템(Virtualization System)과 달리 별도의 호스트 운영체제 없이 서비스만으로 동작 가능한 형태를 의미한다. 이는 개별 어플리케이션 단위로 동작하여 활용할 수 있어 시스템 리소스 활용을 극대화할 수 있는 특성이 있어 윈도우즈, 리눅스(Ubuntu, Fedora, CentOS 등)와 같은 다양한 호스트 운영체제에서 서비스를 제공하고 있다[4]. 그러나 기존 상업용 서버 시스템 및 연구에서는 서버 시스템을 구성하기 위해서 기본 탐재된 호스트 운영체제별 성능에 대한 비교분석을 고려하지 않고 사용하고 있는 실정이다. 이는 디지털 트윈 환경에서 대용량을 활용하는데 있어 직·간접적으로 성능 저하에 원인을 야기할 수 있다.

본 논문에서는 디지털 트윈 환경에서 방대한 양의 센서 데이터를 송수신하기 위한 도커 컨테이너 기반의 서버 시

스템을 구성하기 위한 호스트 운영체제별 성능 비교 분석을 진행하고자 한다. 도커 컨테이너는 어플리케이션 단위를 도커 컨테이너로 관리하는 개념을 도입하여 관리하고, 이를 통해 리소스 경량화에 기여하는 특성이 있다. 본 논문에서는 윈도우즈(서버, 데스크톱), 리눅스(Ubuntu, CentOS, Rocky) 호스트 운영체제에서 데이터셋 별 도커 컨테이너의 성능 비교분석을 진행한다.

이하 본 논문의 구성은 다음과 같다. 2장에서는 배경지식 및 관련 연구에 대해서 서술하고, 실험환경 구성은 3장에서 진행한다. 호스트 운영체제별 도커 컨테이너 성능 비교분석은 4장에서 비교 분석에 대한 상세한 내용을 언급한다. 마지막 5장에서는 본 논문의 결론을 맺는다.

II. Background and Related Works

1. Background

1.1 Virtualization system

가상화는 컴퓨터 리소스를 추상화하는 것을 말한다. 즉 “물리적인 컴퓨터 리소스의 특징을 다른 시스템, 응용 프로그램, 최종 사용자들이 리소스와 상호 작용하는 방식으로부터 감추는 기술”로 정의할 수 있다. 이는 다수의 논리 리소스들을 서버, 호스트 운영체제, 응용프로그램, 저장장치와 같이 하나의 물리 리소스로 만들거나 저장장치, 서버 등과 같은 여러 개의 물리적인 리소스들로 만들 수 있다. 이러한 특성을 가능하게 하는 것을 가상화 시스템이라고 한다[5].

Fig. 1. System architecture of virtualization systems

가상화 시스템은 제공한 하드웨어 플랫폼을 기반으로 제어 프로그램, 즉 호스트 스프트웨어를 통해 실행한다. 호스트 소프트웨어는 호스트 위에 가상화 레이어를 두고, 가상화 레이어에 맞추어 가상머신이라는 컴퓨터 환경을 생성한다. 가상머신은 완전히 독립된 호스트 운영체제 환경을 제공하며, 독립된 하드웨어 플랫폼에 설치된 것처럼 실행할 수 있다.

1.2 Docker container

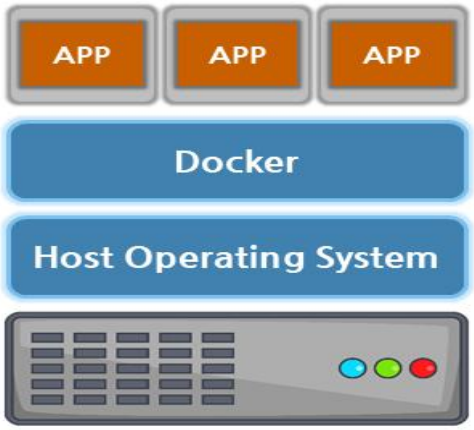


Fig. 2. System architecture of docker container systems

도커 컨테이너는 응용 프로그램들을 프로세스 격리 기술들을 사용하여 실행하고 관리하는 오픈 소스 프로젝트이다[4]. 도커 컨테이너 기능을 인용하자면, “이것은 일종의 소프트웨어를 실행에 필요한 모든 것을 포함하는 완전한 파일 시스템 안에 감싸고 있으며 여기서 코드, 런타임, 시스템 도구, 시스템 라이브러리 등 서버에 설치하는 모든 것을 말한다. 이는 실행중인 어떠한 환경에서도 언제나 동일하게 실행될 것을 보증한다.”라고 정의하고 있다. 도커 컨테이너는 리눅스에서 운영체제 수준 가상화의 추상화 및 자동화 계층을 추가적으로 제공한다.

이러한 도커 컨테이너는 다수의 응용 프로그램, 작업자의 작업, 다른 프로세스들이 자율적으로 하나의 물리 머신이나 여러 개의 가상 머신을 통해 구동할 수 있어 분산 시스템을 생성하는 것을 단순하게 한다. 이는 대용량의 데이터를 저장하고 관리하는 응용 프로그램(Apache Cassandra, 몽고DB, Riak 등)들을 리소스 스케일 업·다운이 용이하다. 또한 도커 컨테이너는 프로세스 작업과 관련하여 워크로드 큐, 기타 분산 시스템들의 생성 및 운영을 단순하게 할 수 있다.

2. Related works

오픈스택(OpenStack)은 서비스형 인프라(Infrastructure as a Service, IaaS) 형태의 클라우드 컴퓨팅 오픈 소스

프로젝트이다[6]. 오픈스택은 비영리 단체인 OpenStack Foundation에서 유지/보수하고 있으며 AMD, 인텔, 캐노니컬, 수세 리눅스, 레드햇, 시스코 시스템즈, 델, HP, IBM, NEC, VM웨어, 야후! 등 150개 이상의 기업이 참여하고 있다. 주요 목적은 리눅스 기반 운영/개발을 진행하고 있으며 프로세싱, 저장공간, 네트워킹의 가용자원을 제어하기 위한 여러 개의 하위 프로젝트로 구성하고 있다. 오픈스택은 커뮤니티에서 6개월 단위로 릴리즈가 이루어지며, 다양한 과정을 통해 지원 및 로드맵을 설정하고 있다.

VMware vSphere는 VMware의 가상화 플랫폼으로 데이터센터를 CPU, 스토리지 및 네트워킹 리소스를 포함하여 집계하는 컴퓨팅 인프라이다. vSphere는 인프라 통합 운영 환경으로 관리하고, 환경에 참여하여 데이터센터를 관리하는 도구를 제공한다[7]. vSphere의 핵심 구성 요소는 ESXi와 vCenter Server로 구성하고 있다. ESXi는 가상 시스템 및 가상 장치를 생성하고 실행하는 가상화 플랫폼이고, vCenter Server는 풀 호스트 리소스 및 네트워크에 연결된 여러 호스트를 관리하는 서비스에 해당한다.

쿠버네티스(Kubernetes)는 도커 컨테이너 화된 어플리케이션의 자동 배치, 스케일링, 운영 등을 자동화하고 관리하는 오픈소스 기반의 프로젝트이다[8]. 이것은 CPU, 메모리, 사용자 지정 지표를 기반으로 어플리케이션을 배치/유지보수하고 스케일인/스케일아웃을 하는 기법을 제공한다. 또한 쿠버네티스는 느슨한 결합으로 설계 되어있어 API에 의한 제공으로 다른 기능을 확장하기 용이하다. 쿠버네티스는 primary, replicaset 구조를 따르며 개개의 노드들의 구성 요소들과 컨트롤 플레인의 일부 구성 요소로 구성하고 관리한다.

도커 컨테이너의 기존 연구에서는 핵심 기술 연구, 성능 개선 등에 초점을 두고 연구를 진행하였다[8-10]. 그러나 기존 연구에서는 성능 개선에 초점을 두었기 때문에 도커 컨테이너를 활용하는 윈도우즈, 리눅스 운영체제에서 성능에 대해 세부판단이 어렵다. 이러한 이유로 인해 최종 사용자는 도커 컨테이너를 운영하는 시스템에서 호스트 운영체제를 선택에 대한 지표가 부족한 실정이다.

III. Experimental Environments

1. Architecture of Docker container

이 절에서는 도커 컨테이너의 구조를 알아본다. 그림 3은 도커 컨테이너의 상세 구조를 보여주고 있다[6].

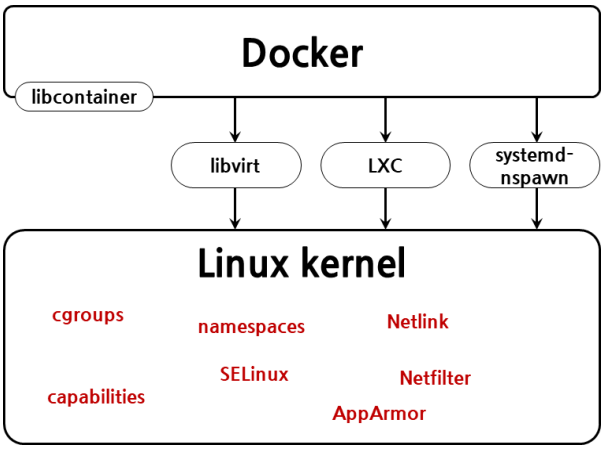


Fig. 3. Docker architecture on linux systems

도커 컨테이너는 경량 프로세스 그룹으로 격리할 수 있도록 하는 가상화 환경이다. 기본적인 구조는 리눅스 커널에서 제공하는 시스템 명령어를 기반으로 동작하는 구조이다. 대표적으로 namespaces는 프로세스 리소스를 분류하고, cgorups은 프로세스 그룹에 대한 리소스 제한한다. 또한 SELiunx 및 Apparmor는 보안을 담당한다. 이렇듯 도커 컨테이너는 최초 리눅스 시스템에서 경량 프로세스를 관리하기 위한 용도로 제안되었다.

최근에는 윈도우즈 10 이후 리눅스를 호환할 수 있는 계층인 리눅스용 윈도우즈 하위 시스템(Windows Subsystem for Linux, WSL)을 제공하고 있다[7]. WSL은 윈도우즈에서 리눅스 실행 파일(Executable and Linkable Format, ELF)을 실행할 수 있는 호환성 계층을 의미한다. WSL는 각 버전 별 특징이 존재하며, 상세 내용은 표 1과 같다.

Table 1. Comparing features of WSL

Feature	WSL1	WSL2
Integration between Windows and Linux	V	V
Fast boot times	V	V
Small resource foot print compared to traditional Virtual Machines	V	V
Runs with current versions of VMware and VirtualBox	V	V
Managed VM		V
Full Linux Kernel		V
Full system call compatibility		V
Performance across OS file systems	V	

표 1과 같이 윈도우즈는 버전 2에 이르러서야 리눅스 커널에 전반을 다 제공하고 있으며, 이를 기반으로 도커 컨테이너를 구동할 수 있는 구조를 제공하고 있다.

그러나 기존 시스템을 구축하고 활용하는 많은 연구자들은 도커 컨테이너 시스템을 구성하기 위해서 호스트 운영체

제별 성능에 대한 정확한 지표가 존재하지 않은 상황이다. 이는 많은 연구자 및 산업에서 도커 컨테이너 호스트 운영체제 시스템을 선택하는데 제약사항이 존재한다. 따라서 본 논문에서는 상호 다른 호스트 운영체제에서 운영 가능한 도커 컨테이너가 실제 운영됨에 따라서 성능에 대한 정확한 지표를 판단할 수 있는 비교 실험을 진행하고자 한다.

2. Experimental environments

실험 환경을 구축하기 위해 이 절에서는 호스트 운영체제인 윈도우즈, 리눅스에서 도커 컨테이너의 시스템 성능을 비교 분석을 진행하고자 한다.

표 2는 호스트 운영체제 및 비교 분석을 진행한 상세 실험 환경을 보여주고 있다. 표 2에서는 윈도우즈 운영체제 3개와 리눅스 운영체제 3개로 구성하였다. 또한 리눅스 운영체제는 CLI와 GUI가 따로 구성되어 있는 경우도 존재하기에 개별 구성요소 별 성능에 대한 비교분석도 추가적으로 진행하였다. 이는 GUI 활용 유무에 따라 성능 차이를 확인하기 위함이다.

Table 2. Experimental environments

Specification			Description
Model Name			NUC11PAHi7
CPU			Intel(R) Core(TM) i7-1165G7 CPU @ 2.80GHz
RAM			Samsung DDR4-3200 16GB
Network			Intel® Ethernet Controller i225-V
GPU			Microsoft basic display dapter
OS	Windows		Windows Desktop 10(OS 1)
			Windows Desktop 11(OS 2)
			Windows Server 2022(OS 3)
	Linux	GUI	CentOS 8.5(OS 4)
			Ubuntu 21.10(OS 5)
			Rocky Linux 8.5(OS 6)
		CLI	CentOS 8.5(OS 7)
			Ubuntu 21.10(OS 8)
			Rocky Linux 8.5(OS 9)

도커 컨테이너의 상세 실험을 비교 분석하기 위해 표 3에서는 리눅스 운영체제를 이용하고 있는 우분투와 CentOS 2개의 컨테이너에 각각 3개 총 6개의 데이터셋을 구성하였다. 개별 운영체제는 웹서버 성능(데이터셋 1,4), 리눅스에서 다중 스레드 성능(데이터셋 2,5), 네트워크 성능(데이터셋 3,6)의 데이터셋들을 활용하여 테스트를 진행하였다. 실험환경에 대한 상세 표본은 표 3의 방법론을 기반으로 실험을 진행하였다.

Table 3. Dataset of experimental environment

Container	Num	Methodology
Ubuntu latest	1	1,000 requests on Apache Bench
	2	100,000 requests on sysbench
	3	1,000 network ping latency
CentOS latest	4	1,000 requests on Apache Bench
	5	100,000 requests on sysbench
	6	1,000 network ping latency

IV. Analysis of Performance Comparison

표 3에서 제시한 데이터셋을 기반으로 그림 4~6에서는 도커 컨테이너를 운영하는 호스트 운영체제의 성능 비교 분석에 대한 결과를 보여주고 있다. 그림 4는 최소 응답시간, 그림 5는 최대 응답시간, 그림 6은 평균 응답시간에 대한 지표이다.

그림 4~6에서 데이터셋 1과 데이터셋 4는 아파치 벤치마크에 1,000 요청 이후 그 결과에 대한 응답시간 성능지표를 보여주고 있다. 실험결과에서 데이터셋 1은 최소 평균 181.45ms, 최대 평균 472.56ms, 전체 평균 220.12ms를 보였으며, 데이터셋 4는 최소 평균 178.78ms, 최대 평균 539.78ms, 전체 평균 223.78ms를 보였다. 리눅스 계열과 윈도우즈 계열의 평균 차이를 비교해보았을 때 리눅스 계열은 1.48% 성능 향상이 있었으며, 윈도우즈 계열은 -2.42%의 성능 하락폭이 있었다.

다음으로 데이터베이스의 응답시간 성능을 비교분석을 진행하기 위해 데이터셋 2와 데이터셋 5는 sysbench를 활용하여 온라인 트랜잭션 처리(Online transaction processing, OLTP)에 대한 데이터베이스 성능지표를 측정했다. 실험결과에서 데이터셋 2는 최소 평균 1.66ms, 최대 평균 37.47ms, 전체 평균 3.33ms를 보였으며, 데이터셋 5는 최소 평균 1.61ms, 최대 평균 43.58ms, 전체 평균 223.78ms를 보였다. 리눅스 계열과 윈도우즈 계열의 평균 차이를 비교해보았을 때 리눅스 계열은 30.55% 성능 향상이 있었으며, 윈도우즈 계열은 -31.61%의 성능 하락폭이 있었다.

마지막으로 데이터셋 4와 데이터셋 6은 가장 기본적으로 탑재한 네트워크 ping 테스트를 진행하였다. 실험결과에서 데이터셋 4는 최소 평균 0.28ms, 최대 평균 1.66ms, 전체 평균 0.8ms를 보였으며, 데이터셋 6은 최소 평균 0.33ms, 최대 평균 1.19ms, 전체 평균 0.82ms를 보였다. 리눅스 계열과 윈도우즈 계열의 평균 차이를 비교해보았을 때 리눅스 계열은 1487.88% 성능 향상이 있었으며, 윈도우즈 계열은 -64.5%의 성능 하락폭이 있었다.

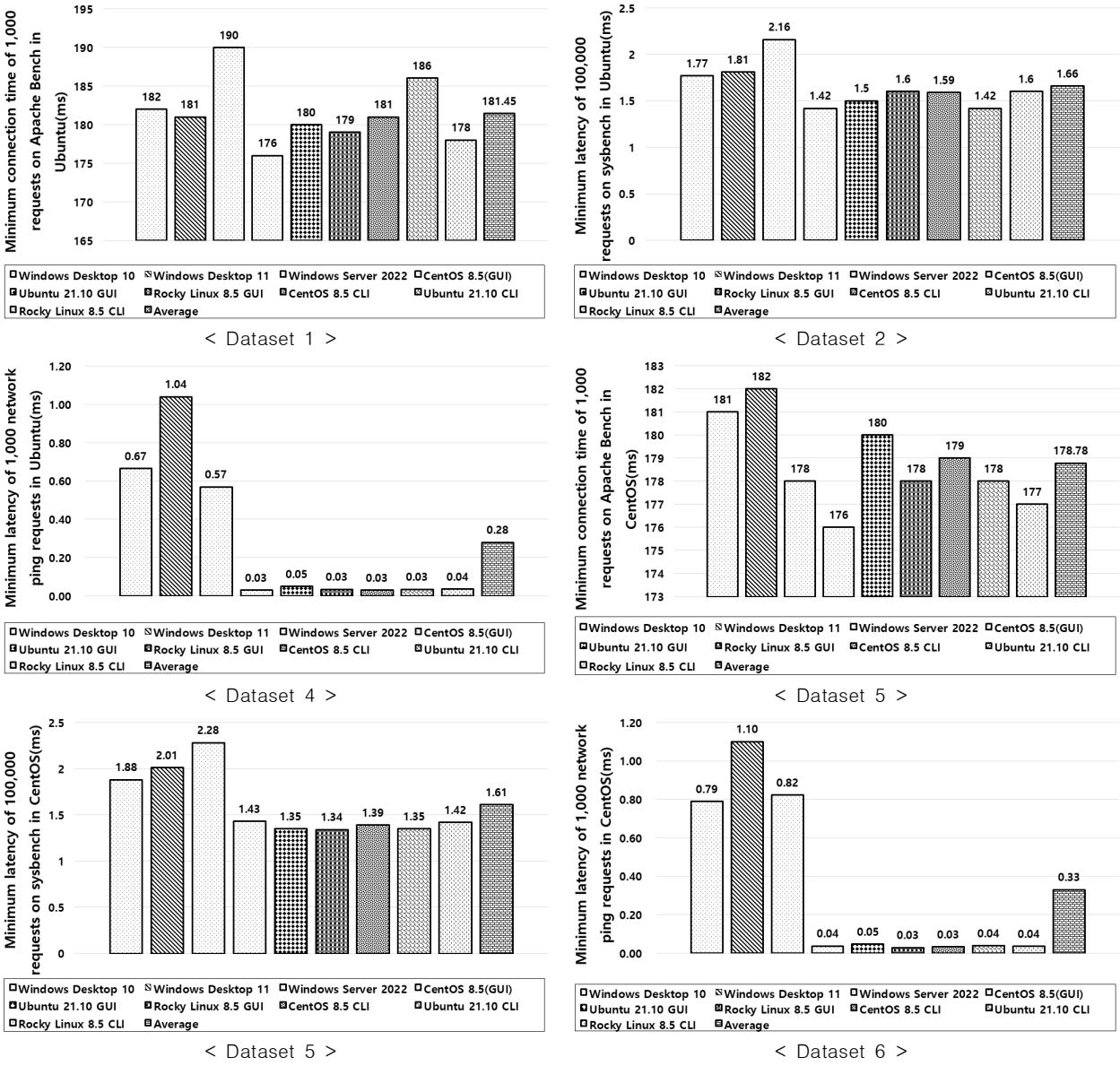


Fig. 4. Experimental results(minimum)

앞서 실험한 결과를 토대로 보았을 때 호스트 운영체제별 결과를 종합하였을 때 도커 컨테이너는 구조적인 특성상 리눅스 계열 대비 윈도우즈 계열에서의 성능이 낮은 것을 실험결과를 통해 확인할 수 있다. 추가적으로 저자들은 리눅스 계열에서의 GUI 포함 유무에 따라 성능 차이도 비교하였으며, 표 4와 5는 해당 지표에 대한 평균 결과이다.

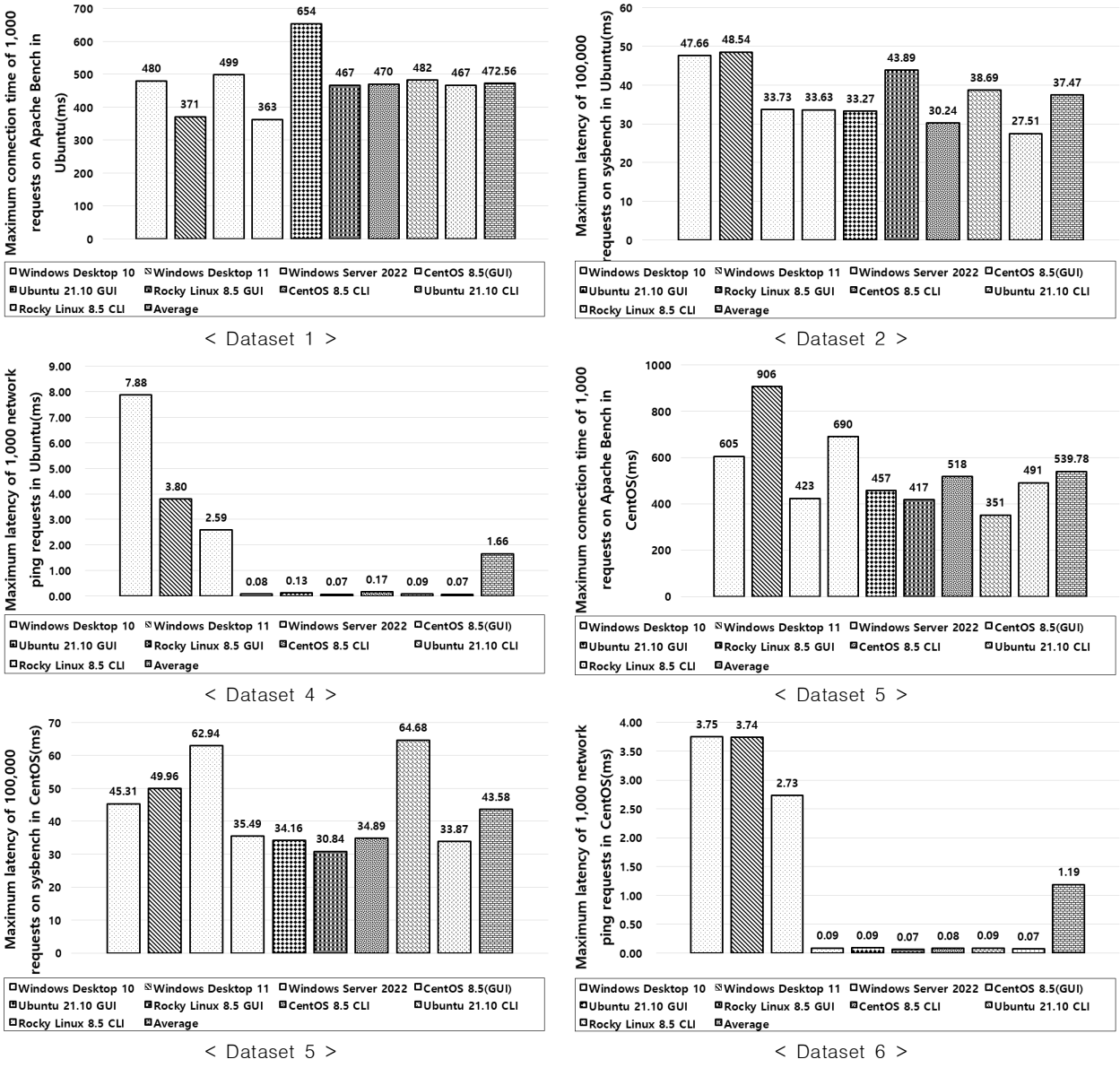
Table 4. Experimental results in linux systems with GUI

OS	dataset					
	1	2	3	4	5	6
4	210	2.59	0.05	215	2.6	0.05
5	212	2.53	0.06	220	2.53	0.06
6	222	2.59	0.04	218	2.61	0.04
total	214.7	2.6	0.1	217.7	2.6	0.1

Table 5. Experimental results in Linux systems with GUI

OS	dataset					
	1	2	3	4	5	6
7	219	2.57	0.05	224	2.56	0.05
8	220	2.45	0.07	208	2.58	0.07
9	219	2.58	0.05	222	2.62	0.05
total	219.3	2.5	0.1	218.0	2.6	0.1

위 결과에서 흥미로운 사실은 리눅스 계열의 운영체제에서 GUI 포함 유무에 따라 응답시간에 대한 성능지표 미비하는 것을 확인할 수 있다. 또한 아파치 벤치마크(데이터셋 2와 데이터셋 4)의 결과에서는 GUI를 포함하고 있는 리눅스 계열 운영체제가 빠른 응답시간을 보였다. 이러한 결과를 보았을 때 리눅스 계열의 호스트 운영체제에서는



< Dataset 1 > < Dataset 2 >

< Dataset 4 > < Dataset 5 >

< Dataset 5 > < Dataset 6 >

Fig. 5. Experimental results(maximum)

도커 컨테이너의 응답시간이 GUI 포함 유무와 무관하다는 것을 실험결과를 통해 확인하였다.

앞서 제시한 비교 분석 내용을 종합하였을 때 도커 컨테이너는 최적화된 서버 시스템을 구성하기에는 리눅스 계열의 운영체제를 구성하는 것이 적합한 것으로 판단된다. 리눅스 계열 중에서도 현재 제시한 세 가지의 호스트 운영체제 기준으로 보았을 때 Ubuntu > CentOS > Rocky Linux 순으로 높은 성능을 보였다. 본 논문에서는 이러한 성능 비교분석을 지표로 구성하였을 때 도커 컨테이너 기반 서버 시스템의 경우 범용적으로 Ubuntu 또는 CentOS를 선택하는 것이 바람직할 것으로 판단된다.

V. Conclusions

본 논문에서는 디지털 트윈 환경에서 방대한 양의 센서 데이터를 송수신하기 위한 도커 컨테이너 기반의 서버 시스템을 구성하기 위한 호스트 운영체제별 성능 비교 분석을 진행하였다. 실험 환경에서는 동일한 환경과 사양을 기반으로 윈도우즈 운영체제 계열과 리눅스 운영체제 계열 6개의 운영체제의 총 9개를 구성하였으며, 6개의 데이터셋을 통해 성능 비교 분석을 진행하였다.

성능 비교 분석 결과에서 본 논문에서는 데이터셋의 전체 평균 대비 윈도우즈 운영체제 계열 평균과 리눅스 운영체제 계열 평균의 차이를 개별 데이터셋 별로 비교하였을 때 전체 평균적으로 3.9%, 62.16% 1552.38%, 7.27%,

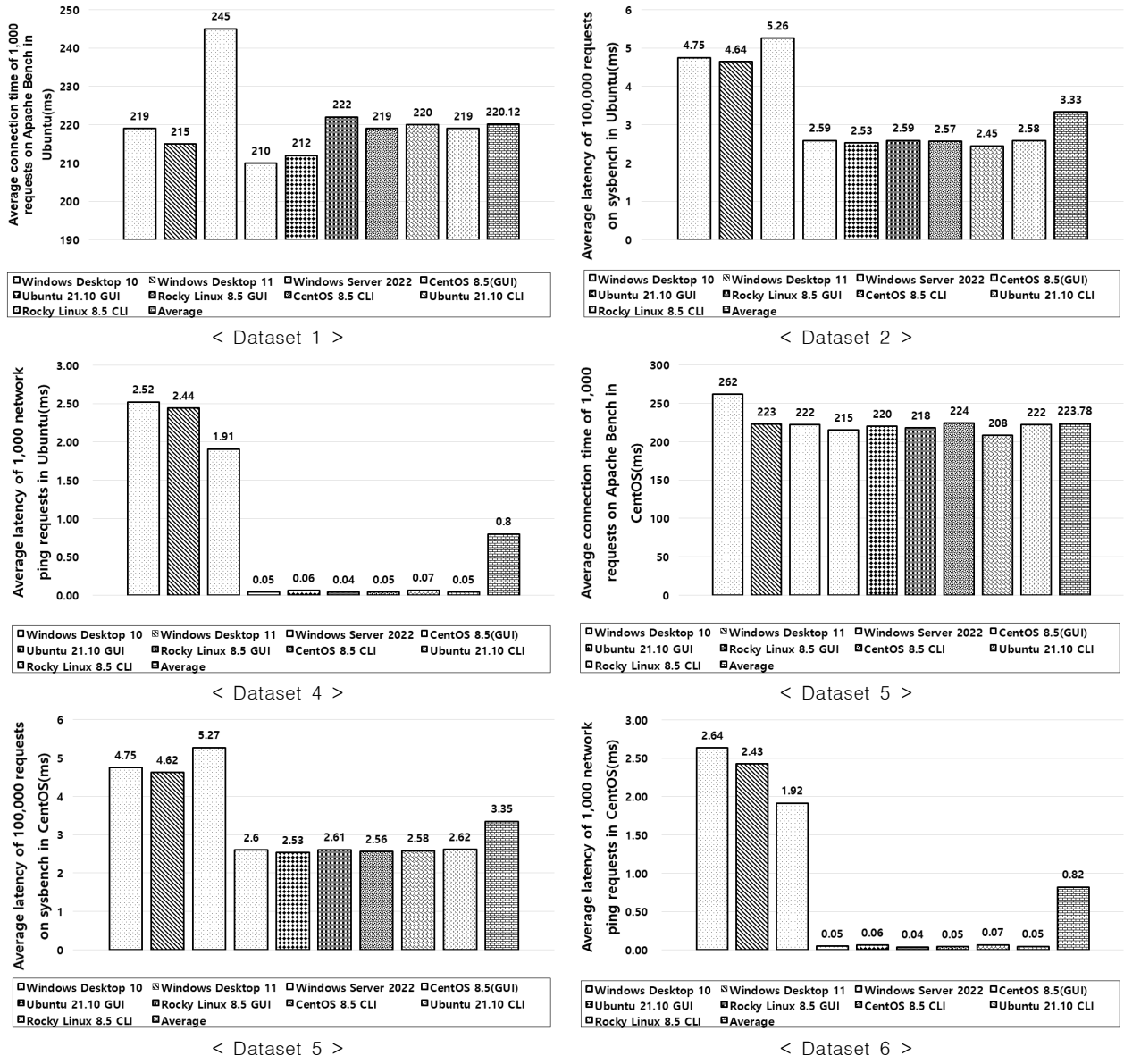


Fig. 6. Experimental results(average)

60.83%, 1567.20%의 리눅스 운영체제 계열이 짧은 지연 시간을 보였음을 실험결과를 통해 확인했다.

향후 연구에서는 윈도우즈 운영체제 계열에서 WSL에 미치는 영향력에 대한 분석을 진행하고 이를 통해 성능적인 향상을 기대할 수 있을지에 대한 비교분석을 진행할 예정이다.

ACKNOWLEDGEMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-02016, Development of a safety management monitoring platform using digital twin).

REFERENCES

[1] L. Junggon, "24,000 safety inspections were conducted at risk-of-death accident sites," http://www.moel.go.kr/news/enews/report/enewsView.do?news_seq=12278

[2] A. Kusiak, "Smart manufacturing," International Journal of Production Research, Vol. 56, No. 1-2, pp. 508-517, 2018. DOI: 10.1080/00207543.2017.1351644

[3] M. Batty, "Digital twins," Environment and Planning B: Urban Analytics and City Science, Vol. 45, No. 5, pp. 817-820. 2018. DOI: 10.1177/2399808318796416

[4] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," Linux journal, Vol. 239, No. 2, pp. 1-5, 2014. DOI: 10.5555/2600239.2600241

[5] M. Pearce, S. Zeadally and R. Hunt, "Virtualization: Issues, security threats, and solutions," ACM Computing Surveys (CSUR), Vol. 45, No. 2, pp. 1-39, 2013. DOI: 10.1145/2431211.2431216

[6] F. Špaček, R. Sohlich, and T. Dulík, "Docker as platform for assignments evaluation," Procedia Engineering, Vol. 100, pp. 1665-1671, 2015. DOI: 10.1016/j.proeng.2015.01.541

[7] H. Barnes, “Pro Windows Subsystem for Linux (WSL),” Apress. pp. 1-321, 2021.

[8] S. Lima, Á. Rocha and L. Roque, "An overview of OpenStack architecture: a message queuing services node. Cluster Computing," Cluster Computing, Vol. 22, No. 3, pp. 7087-7098, 2019. DOI: 10.1007/s10586-017-1034-x

[9] J. Singh, "An Approach to Personalize VMware vSphere Hypervisor (ESXi) Using HPE Image Streamer," In Progress in Advanced Computing and Intelligent Engineering, pp. 363-369, 2021. DOI: 10.1007/978-981-33-4299-6_30

[10] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," IEEE Cloud Computing, Vol. 1, No. 3, pp. 81-84. 2014. DOI: 10.1109/MCC.2014.51

Authors

Sungho Kim received a B.S. degree in the Department of Computer Engineering from Yeungnam University College, Daegu, South Korea in 2012 and a Ph.D. degree in the Department of Computer Engineering from Yeungnam University, Gyeongsan, Gyeongsangbuk, South Korea in 2019. He is currently as a senior researcher in the software research team at Gyeongbuk Institute of IT Convergence Industry Technology (GITC). His current research interests include a big data, a deep learning, embedded systems and non-volatile memory systems.

From 2013 to 2016, Oeon Kwon was a engineer with the Development Center from HanMec IPS, Daegu, South Korea. He received a B.S. degree in the Electronic Engineering from Kyungil University, Gyeongsan, Gyeongsangbuk, South Korea in 2016 and a M.S. degree in the Electronic Engineering from Kyungpook National University, Deagu, South Korea in 2019. During 2019–2020, he was a researcher with System Medical Device Team at Daegu Gyeongbuk Medical Innovation Foundation (DGMIF), Deagu, South Korea. He is currently as a researcher in the software research team at Gyeongbuk Institute of IT Convergence Industry Technology (GITC). His current research interests include a Internet of Things (IoT), a big data and embedded systems.

Jung Han Kim received the Ph.D. degrees in new material Engineering from Mokpo national University, Muangun, South Korea, in 2016 respectively. From 2012 to 2020, He was a researcher in KITECH(Korea Institute of Industrial Technology), Gwangju, South Korea. He is currently as a senior researcher in the Materials and Components research team at Gyeongbuk Institute of IT Convergence Industry Technology (GITC). His research interests include a light metal, a cast, a plastic working and simulation.

JiHyeon Byeon received the Master’s degrees in Mechanical Engineering from Yeungnam University, Gyeongsan, Gyeongsangbuk, South Korea in 2018. He is currently as a researcher in the Materials and Components research team at Gyeongbuk Institute of IT Convergence Industry Technology (GITC). His research interests include a Mechanical structure, material, cutting and simulation.

Sang-Ho Hwang received the B.S., M.S. and Ph.D. degrees in Computer Engineering from Yeungnam University, Gyeongsan, South Korea, in 2009, 2013 and 2017 respectively. From 2017 to 2019, He was a researcher in DGIST (Daegu Gyeongbuk Institute of Science and Technology), Daegu, South Korea. He is currently as a senior researcher in the software research team at Gyeongbuk Institute of IT Convergence Industry Technology (GITC). His research interests include a big data, a deep learning, a data compression, embedded systems and non-volatile memory systems.